Лабораторные работы по курсу «Разработка ПС»

Данный вариант документа не окончателен и может быть дополнен в течение семестра

Преподаватели:

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург
2019

1 Назначение курса

1 Упражнения и лаборатории курса «Разработка ПС» предназначены для практического закрепления навыков, полученных при прослушивании лекционного материала по соответствующему курсу.

2 Общие требования

2.1 Порядок выполнения практической работы

1 Выполнение лабораторной работы начинается с получения задания.
2 Студент должен ознакомиться с заданием на лабораторную работу. В случае если задание непонятно, он может проконсультироваться у преподавателя.
3 Практическая работа выполняется каждым студентом индивидуально. Номер варианта определяется порядковым номером студента в списке группы. Первый в списке получает первый вариант. Если предположить, что вариантов шесть, то седьмой студент по списку получает снова первый вариант. Студент не может выбрать другой вариант.
4 После получения и согласования с преподавателем задания на лабораторную работу студент приступает к выполнению теоретической части работы. В ходе теоретической части работы студент разрабатывает:
   - требования к программному продукту;
   - математические методы и алгоритмы;
   - структуру программы;
   - форматы представления данных.
5 После разработки теоретической части студент представляет преподавателю разработанный материал, и, получив разрешение преподавателя, приступает к непосредственному кодированию разработанной программы на компьютере.
6 Написав программу, студент должен её тщательно протестировать. После этого он должен сдать программу преподавателю, продемонстрировав её работу и исходный код.
7 По завершении работы студент оформляет отчёт, к которому прилагается исходный код программы.

2.2 Требования к программному коду

1 Все идентификаторы (названия переменных, функций, классов и модулей) должны отражать назначение именуемых объектов. Все называния должны быть даны на английском языке. Не допускается использование транслитерации и других нестандартных приёмов именования.
2 Все константные литералы в коде должны быть объявлены как константы (const).
3 Стиль оформления кода (отступы, именование переменных, функций, классов и т.д.) должен последовательно соблюдаться в работе и соответствовать одному из общепринятых стандартов (например, C# Naming Guidelines).

2.3 Требования к программе

1 Программа при запуске должна выводить подробную информацию о назначении программы, авторе, решаемой задаче и предоставляемых результатах. В случае если программа выполнена с использованием графического интерфейса пользователя (приложение Windows) – программа должна выводить соответствующую информацию при запуске и в ответ на выбор соответствующего пункта меню. В этом случае должна быть возможность отключить вывод приветствия при запуске программы.
2 Выполнение консольной программы должно быть закольцовано. Завершение работы программы должно производиться только при выборе соответствующего пункта меню.

2.4 Общие требования к работам

1 Все ошибки, которые могут возникнуть в программе должны быть обработаны с выдачей соответствующих диагностических сообщений. Метод обработки ошибок должен быть единым для всей программы.
2 Интерфейс пользователя должен быть полностью отделён от вычислительных процедур программы.
3 Все интерфейсы должны быть документированы.

2.4.1 Работа с файлами

1 Во всех работах кроме консольного ввода-вывода (экран, клавиатура) необходимо предоставить возможность:
   - вводить исходные данные из файла;
   - сохранять исходные данные в файле;
   - сохранять результат работы программе в файле.
2 Полный путь к файлам в каждом случае задает пользователь.

2.4.2 Тестирование

1 Во всех работах должны присутствовать модульные тесты.

2.5 Содержание отчёта

1 В отчёт входит описание:
   - постановки задачи;
   - исходных данных;
   - особых ситуаций;
   - математических методов и алгоритмов решения задачи;
   - форматов представления данных в памяти и на внешних носителях;
   - структуры программы;
   - модулей, функций и переменных программы;
   - блок-схем алгоритмов программы;
   - хода выполнения работы;
   - полученных результатов.
2 Отчёт оформляется на листах формата А4 с обязательным титульным листом, на котором указываются название работы; ФИО исполнителя; ФИО преподавателей и т.д.

2.5.1 Постановка задачи

1 Постановка задачи указывает, какая цель должна быть достигнута при разработке программы. Какую задачу должна решать программа, и в каких условиях будет функционировать.

2.5.2 Исходные данные

1 Исходными данными являются любые данные, которые программа получает для обработки.
2 Описание исходных данных должно содержать:
   - семантику (назначение) данных;
   - единицы изменения;
   - представление в программе.

2.5.3 Особые ситуации

1 Под особыми ситуациями понимаются ситуации, в которых поведение программы может не соответствовать поведению, ожидаемому пользователем.

2 Все особые ситуации должны быть описаны и соответствующим образом обработаны в программе.
3 Примерами особых ситуаций являются:
   - некорректный ввод;
   - отсутствие ожидаемых программой файлов;
   - возможное деление на ноль в ходе вычислений;
   - нехватка оперативной памяти.

2.5.4 Математические методы и алгоритмы решения задач

1 Все используемые программой алгоритмы и математические методы решения задач должны быть описаны в специальном разделе в форме и полноте, достаточной для восприятия другими разработчиками.

2.5.5 Форматы представления данных

1 Для всех пользовательских типов данных (не являющихся частью языка) должны быть документированы назначение и мотивация выбора конкретного типа данных.
2 Должны быть документированы форматы всех внешних ресурсов. Структура данных, сохраняемых в файлах и т.д.

2.5.6 Структура программы

1 Разработанная структура программы (разделение на модули, интерфейсы, шаблоны проектирования) должна быть документирована.
2 Должна быть описана основная последовательность работы программы (вызова функций, методов и т.д.).
3 Все модули, функции, методы и пользовательские типы данных должны быть соответствующим образом документированы в отчёте.

2.5.7 Блок-схемы алгоритмов программы

1 Построение блок-схем алгоритмов регламентируется ГОСТ 19.701-90 (ИСО 5807-85) «Единая система программной документации. Схемы алгоритмов программ, данных и систем. Условные обозначения и правила выполнения».

2.5.8 Описание хода выполнения лабораторной работы

1 В отчёте должно содержаться подробное описание хода выполнения лабораторной работы.
2 Основное внимание должно быть уделено выполнению работы за компьютером.
3 Должны быть описаны все новые методы и приёмы, использованные в ходе выполнения лабораторной работы. Таковыми могут быть:
   - создание проекта и запуск программы;
   - работа с отладчиком (точки останова и протоколирования, просмотр значений переменных);
   - поиск ошибок в программе;
   - работа с устройствами ввода-вывода.
4 Из отчёта должно быть ясно:
   - как выполнялась лабораторная работа;
   - какие были проблемы и как они были решены;
   - какие проблемы остались нерешёнными;
   - какие моменты были или остались непонятными.

2.5.9 Результаты работы программы

1 Необходимо указать, какие результаты производит программа.
2 Необходимо указать в каком формате пользователь получает результат.
3 Должно быть приведено три варианта результатов выполнения программы с различными исходными данными.

2.5.10 Исходный текст программы

1 Исходный текст программы распечатывается и прилагается к отчёту.

2.5.11 Документирование и комментирование исходного текста

1 Все пользовательские типы данных должны быть прокомментированы.
2 Все функции, классы и модули должны быть прокомментированы.
3 Каждый модуль должен начинаться с комментария, указывающего его назначение, автора, используемые алгоритмы.
4 Каждая нетривиальная функция должна предваряться комментарием, описывающим:
   - назначение;
   - входные данные;
   - результаты.
5 В функциях, где соответствующее описание будет полезным, также следует описать:
   - предусловия;
   - постусловия;
   - инварианты.

2.6 Защита и сдача лабораторной работы

1 В весеннем семестре выполняются шесть лабораторных работ из данного документа. График сдачи лабораторных работ:

№	Представление работы преподавателю	Окончательная защита
1	февраль	март
2	март	март
3	март	апрель
4	апрель	апрель
5	апрель	май
6	май	май

2 Лабораторная работа защищается преподавателям, ведущим лабораторные и практические работы.
3 Для защиты необходимо иметь отчёт о проделанной работе и продемонстрировать работоспособную программу.
4 Подпись за лабораторную работу выставляет преподаватель, которому работа была защищена.
5 Окончательная сдача лабораторных работ является допуском к экзамену (или зачёту) по предмету.

2.6.1 Окончательная сдача лабораторных работ

1 Для сдачи лабораторных работ, необходимо представить:
   - комплект отчётов по лабораторным работам;
   - папку с исходными кодами программ и выполняемыми модулями.
2 В папке должен содержаться файл readme.txt, в котором должно быть указано:

- Ф.И.О. выполнившего работы
- Год, название предмета, названия работ.

2.7 Пример выполнения задания

При выполнении лабораторной работы будет использоваться компилятор Microsoft Visual Studio 2015.

2.7.1 Постановка задачи

Методом Монте-Карло вычислить число «пи».

2.7.2 Исходные данные

В качестве исходных данных программа использует вводимое пользователем число испытаний при проведении эксперимента.

2.7.3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации.

- Если пользователь ввёл число испытаний меньше одного, то эксперимент провести невозможно.
- Если пользователь ввёл число испытаний меньше разумного предела для проведения статистического эксперимента, результаты могут быть недостоверными.
- Если пользователь ввёл очень большое число испытаний, то эксперимент может занять значительное время, о чём пользователь должен быть предупреждён.

2.7.4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будет использован метод Монте-Карло, который заключается в случайном выборе координат точек внутри квадрата заданного размера.

Для каждой точки будет проверяться попадание во вписанную в квадрат окружность. Таким образом, по окончании эксперимента мы будем располагать двумя числами:

- N – число случайно выбранных точек;
- M – число точек, попавших внутрь вписанной окружности.

Поскольку нам известна площадь квадрата и площадь вписанной окружности, то мы можем вычислить отношение площадей этих фигур. Если принять сторону квадрата за единицу, то его площадь будет равна одной квадратной единице.

Площадь круга, вписанного в этот квадрат, может быть определена по формуле:

$$S = \pi r^2 = \frac{\pi d^2}{4} \quad (1)$$

Таким образом, число π можно выразить через площадь и диаметр вписанного круга следующим образом:

$$\pi = 4\frac{S}{d^2} \quad (2)$$

Площадь вписанного круга можно найти из отношения M к N. Они относятся друг к другу так, как относится площадь вписанной окружности к площади квадрата. А площадь квадрата нам известна – она единична. Исходя из этого, получаем полную формулу для вычисления числа π по известным нам числам M и N.

$$\pi = \frac{4}{d^2} \cdot \frac{M}{N} \quad (3)$$

Все величины в данной формуле нам известны. Однако, как было указано выше, M является статистической величиной, которая будет рассчитана в результате проверки попадания случайных точек в окружность. Для генерации набора точек и проверки их попадания в круг необходимо написать программу.

2.7.5 Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

Имя	Тип	Описание
tries	int	Число попыток в эксперименте
inCircle	int	Число точек, попавших в окружность

Для задания минимального и максимального пределов числа экспериментов используются следующие константы

Таблица 2 – Константы, используемые в программе

Имя	Тип	Значение	Описание
LowerLimit	const int	100	Минимальное число экспериментов, обеспечивающих достоверность
UpperLimit	const int	10000000	Число экспериментов, при превышении которого вычисления могут быть долгими

Для задания координат точки на плоскости используется структура *Point2D*, в которой задаются x и y координаты точки.

2.7.6 Структура программы

В силу своей простоты программа помещена в одном исполняемом модуле. Программа разделена на несколько функций:

Таблица 3 – Функции, составляющие программу

Имя	Описание
GeneratePoint	Создание точки в заданных координатах
IsInsideCircle	Проверка на нахождение точки внутри окружности
ProcessInput	Обработка ввода пользователя

2.7.7 Описание хода выполнения лабораторной работы

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio 2015. В нём был создан проект.
2. В начальном варианте программы было перепутано условие выхода из цикла `while`. Было указано:

```
while (InputSuccess) {
```

в результате чего тело цикла ни разу не выполнялось. Для выявления проблемы было использовано пошаговое выполнение программы, в ходе которого выяснилась причина ошибки, и программа была исправлена.

2.7.8 Результаты работы программы

В результате вычислений программа выводит примерное значение числа «пи». Точность вычислений определяется числом проведённых экспериментов и качеством используемого генератора случайных чисел.

2.7.9 Исходный текст программы

```
[ Начало программы ---]
// Lab1.cpp
// Лабораторная работа №0.
// Использование языка C# для математических расчётов
// Расчёт числа "пи" методом Монте-Карло
// Студент группы NNN, Фамилия Имя Отчество. 2018 год
using System;


namespace Method_Monte_Carlo
{
  class Program
  {
    static void Main(string[] args)
    {
      int inCircle = 0;
      int tries = 0;
      tries = ProcessInput();

      var rand = new Random();

      // Генерация и проверка точек
      for (int i = 1; i <= tries; ++i)
      {
        Point2D pt = GeneratePoint(rand);
        if (IsInsideCircle(pt))
          ++inCircle;
      }

      // Вычисление числа "пи"
      Console.Write("Число пи = ");
      Console.Write(4 * (inCircle / (double)tries));

      Console.ReadKey();
    }


    /// <summary>
    /// Генерация случайной точки с координатами x и y в
    ///интервале от нуля до единицы
    /// </summary>
    static Point2D GeneratePoint(Random rand)
    {
      Point2D pt = new Point2D
      {
        X = rand.NextDouble(),
        Y = rand.NextDouble(),
      };
```

```
  return pt;
}



/// <summary>
/// Проверка нахождения точки внутри окружности
/// </summary>
static bool IsInsideCircle(Point2D pt)
{
  const double circleRadius = 0.5;
  Point2D circle = new Point2D
  {
    X = 0.5,
    Y = 0.5,
  };

  double distFromCenter = Math.Sqrt(
    Math.Pow(circle.X - pt.X, 2) +
    Math.Pow(circle.Y - pt.Y, 2)
  );

  return distFromCenter < circleRadius;
}

/// <summary>
/// Обработка ввода пользователя
/// <summary>
static int ProcessInput()
{
  const int LowerLimit = 100;
  const int UpperLimit = 10000000;
  bool inputSuccess = false;
  int tries = 0;

  while (!inputSuccess)
  {
    Console.Write("Введите число экспериментов:");
    string consoleInput = Console.ReadLine();

    if (!int.TryParse(consoleInput, out tries))
    {
      Console.WriteLine(
        "Число экспериментов должно быть числом.");
      continue;
    }
    if (tries <= 0)
    {
      Console.WriteLine(
        "Число экспериментов должно быть положительным.");
      continue;
    }
```

```
        inputSuccess = true;
      }
      if (tries < LowerLimit)
      {
        Console.WriteLine("Результат может быть неточным.");
      }
      if (tries > UpperLimit)
      {
        Console.WriteLine("Вычисления могут быть длительными.");
      }
      return tries;
    }

    /// <summary>
    /// Структура, описывающая точку на плоскости.
    /// </summary>
    private struct Point2D
    {
      public double X, Y;
    };
  }
}
[--- Конец программы.]
```

3 Лабораторная работа №1. Использование массивов

Необходимо выделить массив требуемого размера, запросив его у пользователя при запуске программы или считав из файла. В программе должны быть предусмотрены три варианта заполнения исходного массива: пользователем с клавиатуры, из файла и случайными числами.

3.1 Варианты заданий

1 Задан одномерный массив размером N. Определить количество отрицательных чисел, количество положительных чисел и среднее арифметическое всех чисел.
2 Задан одномерный массив размером N. Сформировать 2 массива размером N/2, включая в первый элементы исходного массива с четными индексами, а во второй — с нечетными. Вычислить суммы элементов каждого из массивов.
3 Определить число отрицательных элементов, расположенных перед наибольшим положительным элементом одномерного массива, размер которого M.
4 В заданном одномерном массиве поменять местами первый и последний положительные элементы.
5 Написать программу для вычисления среднего арифметического значения всех элементов массива. Для отрицательных элементов использовать их абсолютные значения.
6 Написать программу для поиска в массиве наибольшего элемента.
7 Написать программу по упорядочению элементов заданного массива в следующем порядке: сначала идут положительные числа, потом — нули и в конце — отрицательные.
8 Написать программу для сортировки массива по возрастанию.
9 Написать программу для циклического сдвига массива вправо на N позиций.
10 В заданном массиве найти число элементов, расположенных после его наибольшего отрицательного элемента.

4 Лабораторная работа №2. Знакомство с наследованием и интерфейсами

Студент выполняет задание обычной или повышенной сложности

Создать интерфейс ICipher, который определяет методы поддержки шифрования строк. В интерфейсе объявляются два метода Encode() и Decode(), которые используются для шифрования и дешифрования строк, соответственно. Реализовать 2 класса реализующих данный интерфейс. Один из алгоритмов в соответствие с вариантом, второй по выбору.

4.1 Варианты заданий обычной сложности

1 Шифр Цезаря.
2 Гаммирование.
3 Атбаш.
4 Аффинный шифр.
5 ROT13.
6 Шифр Виженера.
7 Шифр Бофора.
8 Шифр Плейфера.
9 Шифр Хилла.

4.2 Варианты заданий повышенной сложности

1 AES.
2 ГОСТ 28147-89.
3 DES.
4 3DES.
5 RC2 (Шифр Ривеста (Rivest Cipher или Ron’s Cipher)).
6 RC5.
7 Blowfish.
8 Twofish.
9 NUSH.
10 IDEA (International Data Encryption Algorithm, международный алгоритм шифрования данных).
11 CAST (по инициалам разработчиков Carlisle Adams и Stafford Tavares).
12 3-WAY.
13 Khufu и Khafre.
14 Kuznechik.

5 Лабораторная работа №3. Знакомство с WinForms

Студент выполняет задание обычной или повышенной сложности.

Необходимо написать приложение с использованием технологии WinForms для построения графика функции и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты (при их наличии). При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение.

5.1 Варианты заданий обычной сложности

1. Верзьера Аньези.

$$y = \frac{a^3}{a^2 + x^2}$$

2. Декартов лист

$$y = \pm x\sqrt{\frac{l + x}{l - 3x}}, \text{ где } l = \frac{3a}{\sqrt{2}}$$

3. Овал Кассини

$$y = \pm\sqrt{\sqrt{a^4 + 4c^2x^2} - x^2 - c^2}$$

4. Лемниската Бернулли

$$y = \pm\sqrt{\sqrt{c^4 + 4x^2c^2} - x^2 - c^2}$$

5. Синусоида

$$y = a + b\sin(cx + d).$$

6. Циклоида

$$x = r\arccos\frac{r - y}{r} - \sqrt{2ry - y^2}$$

7. Трактриса

$$x = \pm\left(a\ln\frac{a + \sqrt{a^2 - y^2}}{y} - \sqrt{a^2 - y^2}\right), \text{ при } y \in (0, a)$$

8. Цепная линия

$$y = \frac{a}{2}(e^{x/a} + e^{-x/a})$$

9. Кубика Чирнгауза

$$27ay^2 = (a - x)(8a + x)^2$$

10. Прямая строфоида

$$y = \pm x\sqrt{\frac{a + x}{a - x}}.$$

11. Циссоида Диокла

$$y^2 = \frac{x^3}{2a - x}.$$

12. Лемниската Жероно

$$y = \pm\sqrt{\frac{a^2x^2 - x^4}{a^2}}.$$

13. Астроида

$$x^{2/3} + y^{2/3} = R^{2/3}$$

14. Квадратриса

$$x = y\ \mathrm{ctg}\,\frac{\pi y}{2R}$$

5.2 Варианты заданий повышенной сложности

Необходимо выполнить задание из вариантов обычной сложности добавив в программу возможность экспорта исходных данных и результатов расчета в MS Excell

6 Лабораторная работа №4.

Студент выполняет задание обычной или повышенной сложности.
Необходимо написать приложение с использованием технологии WinForms реализующие вариант задания. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

6.1 Варианты заданий обычной сложности

Необходимо написать программу
1. Хранения списка банковских счетов.
2. Хранения списка студентов.
3. Хранения расписания поездов.
4. Хранения списка товаров в магазине.
5. Хранения списка должников.
6. Хранения естественных астрономических объектов (звезды, планеты, галактики).
7. Хранения искусственных астрономических объектов (искусственные спутники земли, межпланетные станции).
8. Хранения списка книг в библиотеки.
9. Хранения фильмотеки.
10. Хранения результатов спортивных соревнований.

6.2 Варианты заданий повышенной сложности

Необходимо выполнить задание из вариантов обычной сложности используя паттерн MVP (или MVVM если используется WPF) и библиотеку Autofac для внедрения зависимостей.

7 Лабораторная работа №5.

Пункт в разработке.

8 Лабораторная работа №6.

Пункт в разработке.